ZABBIX '26
CONFERENCE
GERMANY

Hilbig
IT WORKS

Virtual Control Room

A Modular Platform for Traffic, Event, Logistic and Agricultural Management Based on Zabbix

Berlin, April 29 - 30

The Challenge

From Monitoring to Orchestration

Monitoring vs. Control

The disconnect between monitoring and control creates an "air gap," making it difficult for operators to manage systems effectively without a unified interface for real-time data and actions.

Data Fragmentation

Multiple data sources lead to fragmentation, complicating the creation of a cohesive "situation picture" needed for operators to make informed decisions and maintain operational efficiency.

Why Zabbix?

Zabbix: The Heart of VCR

The foundation

- Why rebuild monitoring? Zabbix does it best.
- We treat IoT devices as "Hosts" and their telemetry as "Items".
- The history function is crucial for auditing automated control actions.

The value proposition

- Reliable Data Ingestion: Telemetry from thousands of sensors.
- Powerful Event Detection: Triggers are the "nerves" of our control logic.
- Portal Base: Built-in user management and dashboarding.
- History as Audit Trail: Trapper items act as a "Flight Recorder"

Transition: How does this all fit together architecturally?

What actors and integrations do we have?

The VCR System Context

Actors and Use Cases

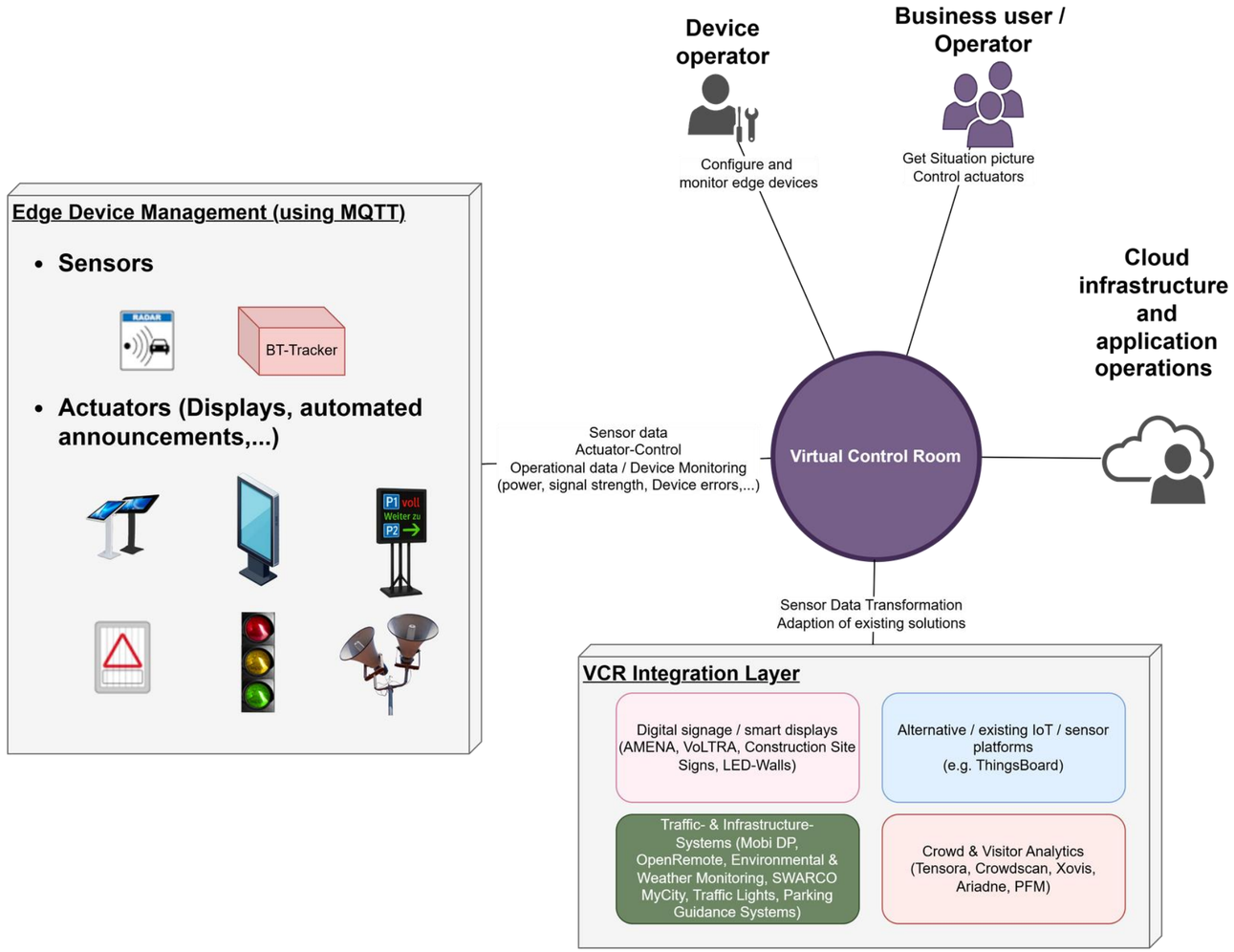
- **Business users**
  - Get shared situation picture
  - Detect issues and operational risks
  - Coordinate response and escalation
  - Trigger Actions and Review Outcomes
- **Device operators**
  - Configure and monitor edge devices (voltage, connectivity, physical position,...)

Edge Device Management

- Connecting field devices (Radar, LED, GPS) via MQTT to Zabbix.
- After some stability problems with the builtin Zabbix Agent2 MQTT we created an own MQTT Broker for fast, lightweight communication

Integration Layer

- Using Apache Camel for legacy protocols.



How does it work and how is Zabbix extended?

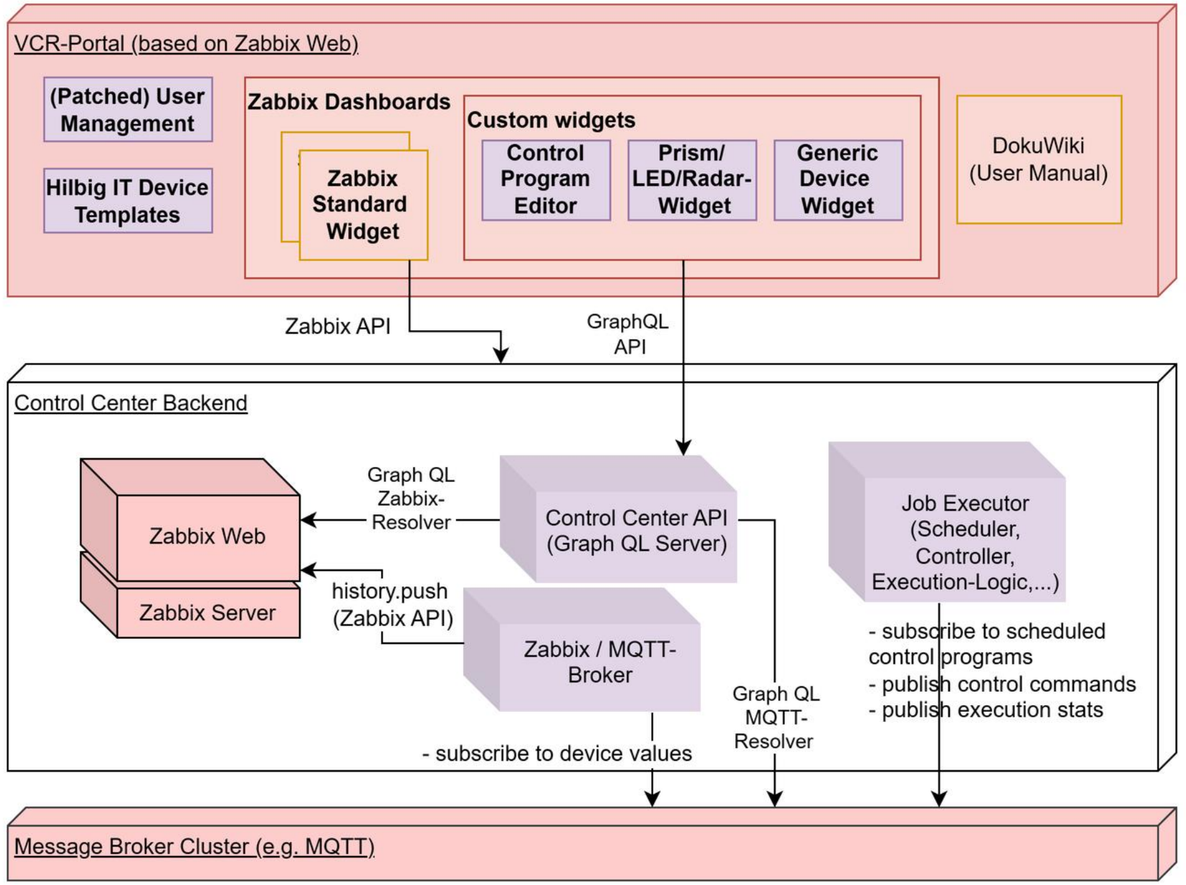
A modular set of microservices extending Zabbix

Node.JS Backend

- Job Executor for running the "Control Programs", taking MQTT as “single source of truth”
- GraphQL DSL: Wrapping Zabbix API to speak "Device Language" instead of "Host/Item Language" in a modern, AI / MCP friendly interface for our web components.

Angular Frontend Extensions

- Embedding Angular Webcomponents as native-feeling widgets.



Injecting rich, ineractive Angular Webcomponents into sysmaps

Extension Magic Deep Dive 1: Hacking the sysmap SVG DOM

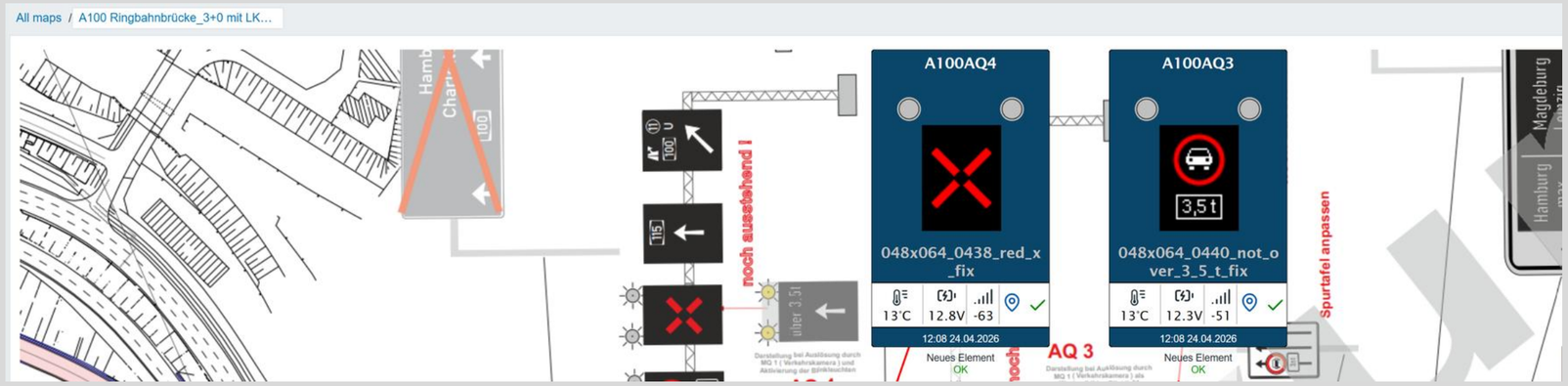
1. Lookup sysmap container elements and svg child by **class=”sysmap-widget-container”**

```
<div class="sysmap-widget-container">
  <div class="sysmap-caption">A6 FDE Wolpertshausen 2023 Fa
    Nürnberg km 681+750 bis km 687+800, 2 of 11 elements in
    Host, Status problem, 00 (MQ1) RADAR01
    00000000-0000-0000-0000-726164617231, 3 problems. Host,
    00000000-0000-0000-0000-726164617232, 3 problems. Host,
    00000000-0000-0000-0000-707269736d32. Host, Status ok,
    00000000-0000-0000-0000-707269736d33. Image, Status ok,
    x. Image, Status ok, . Host, Status ok, 01 (AQ1) PRISM0
    00000000-0000-0000-0000-707269736d31. Image, Status ok,
  </div><svg viewBox="0 0 1920 1080" style="..."
    preserveAspectRatio="xMinYMin meet">
    <g class="shadow-buffer" style="..."></g>
    <g class="map-container" font-family="&quot;Trebuchet M
```

2. Remove the svg-Element and create a selector element for calling the Angular Webcomponent instead, passing selements metadata

```
<svg id="mySVG" viewBox="0 0 1920 1080" style="max-width: 1920px; max-height: 1080px;
  " preserveAspectRatio="xMinYMin meet"> == $0
  <g class="shadow-buffer" style="visibility: hidden;"></g>
  <g class="map-container" font-family="&quot;Trebuchet MS&quot;, Helvetica, sans-ser
  if" font-size="10px">
    <g class="map-background" fill="#FFFFFF">…</g>
    <g class="map-grid" stroke="#CCD5D9" fill="#CCD5D9" stroke-width="1" stroke-
    dasharray="4,4" shape-rendering="crispEdges"></g>
    <g class="map-shapes"></g>
    <g class="map-elements">…</g>
    <g class="map-elements">
      <g app-road-device-svg-widget _nghost-ng-c1736990859>…</g>
      <g app-road-device-svg-widget _nghost-ng-c1736990859>
        <svg _ngcontent-ng-c1736990859 viewBox="0 0 180 300" x="1100" y="0" width="18
        0" height="300">
          <g _ngcontent-ng-c1736990859 transform="translate(0,0)">
            <rect _ngcontent-ng-c1736990859 width="100%" rx="4" ry="4" height="300"
            class="roadSignDisplayBackground_prism_display"></rect>
            <g _ngcontent-ng-c1736990859 class="roadSignDisplay">
              <g _ngcontent-ng-c1736990859 data-menu-popup="{"type":"map_element","dat
              a":{"sysmapid":"29","selementid":"130","unique_id":"U000001","selementty
              pe":"0","selementsubtype":"0","selementlabel":"Neues Element","selementi
```

3. Enjoy interactive, modern angular webcomponent directly embedded into existing Zabbix Sysmap functionality



Wrapping Zabbix API to speak "Device Language" instead of "Host/Item Language"

Extension Magic Deep Dive 2: The GraphQL-API

1. Define the schema using GraphQL

```
"""
DistanceTracker represents a device
which can detect other devices around
itself and estimate the distances, e.g.
by using Bluetooth scanning technology
and estimating the distance.
"""
type DistanceTrackerDevice implements
 Host & Device {
    """Internal Zabbix ID"""
    hostid: ID!
    """List of host groups
    this device belongs to."""
    hostgroups: [HostGroup!]
    """Device configuration tags."""
    tags: DeviceConfig
    """State of the device."""
    state: DistanceTrackerState
}
```

2. Define current device state as trapper and dependant items - optionally using the GraphQL MCP Server and an agent (our broker uses the key to map to topics and uses history.push)

Name ▲	Key	Type
MQTT_COUNT: count	state.current.values.count	Dependent item
MQTT_COUNT	mqtt.trap[deviceValue/count]	Zabbix trapper
MQTT_DISTANCE	mqtt.trap[deviceValue/distance]	Zabbix trapper
MQTT_NAME	mqtt.trap[deviceValue/name]	Zabbix trapper
MQTT_SERVICE_DATA	mqtt.trap[deviceValue/ServiceData]	Zabbix trapper
MQTT_COUNT: MQTT_STATE	currentstate	Dependent item
MQTT_COUNT: timeFrom	state.current.values.timeFrom	Dependent item
MQTT_COUNT: timeUntil	state.current.values.timeUntil	Dependent item

3. Define device meta data as tags. JSON is supported (limited by max. Zabbix tag value length)

Tags 15 | Macros | Value mapping

Tags	Name	Value
	automatismUsage.json_command.condition values	[{"name": "count","displayName":"Anzahl","topicName":"count"}]
	class	roadwork
	deviceType	bttracker
	deviceWidgetPreview.BOTTOM_LEFT.key	timeFrom
	deviceWidgetPreview.BOTTOM_LEFT.unit	Startzeit
	deviceWidgetPreview.BOTTOM_LEFT.unit_font_size	8

4. The generic GraphQL-Resolver maps item + history using the Device DSL using schema introspection and Zabbix API without extra code

```
type DistanceTrackerValues {
    """
    Number of unique device keys detected
    """
    count: Int
    """
    Detailed information about devices.
    """
    distances: [SensorDistanceValue!]
}
```

```
type WidgetPreview {
    """Top-left field specification."""
    TOP_LEFT: DisplayFieldSpec
    """Top-right field specification."""
    TOP_RIGHT: DisplayFieldSpec
    """Bottom-left field specification."""
    BOTTOM_LEFT: DisplayFieldSpec
    """Bottom-right field specification."""
    BOTTOM_RIGHT: DisplayFieldSpec
}
```

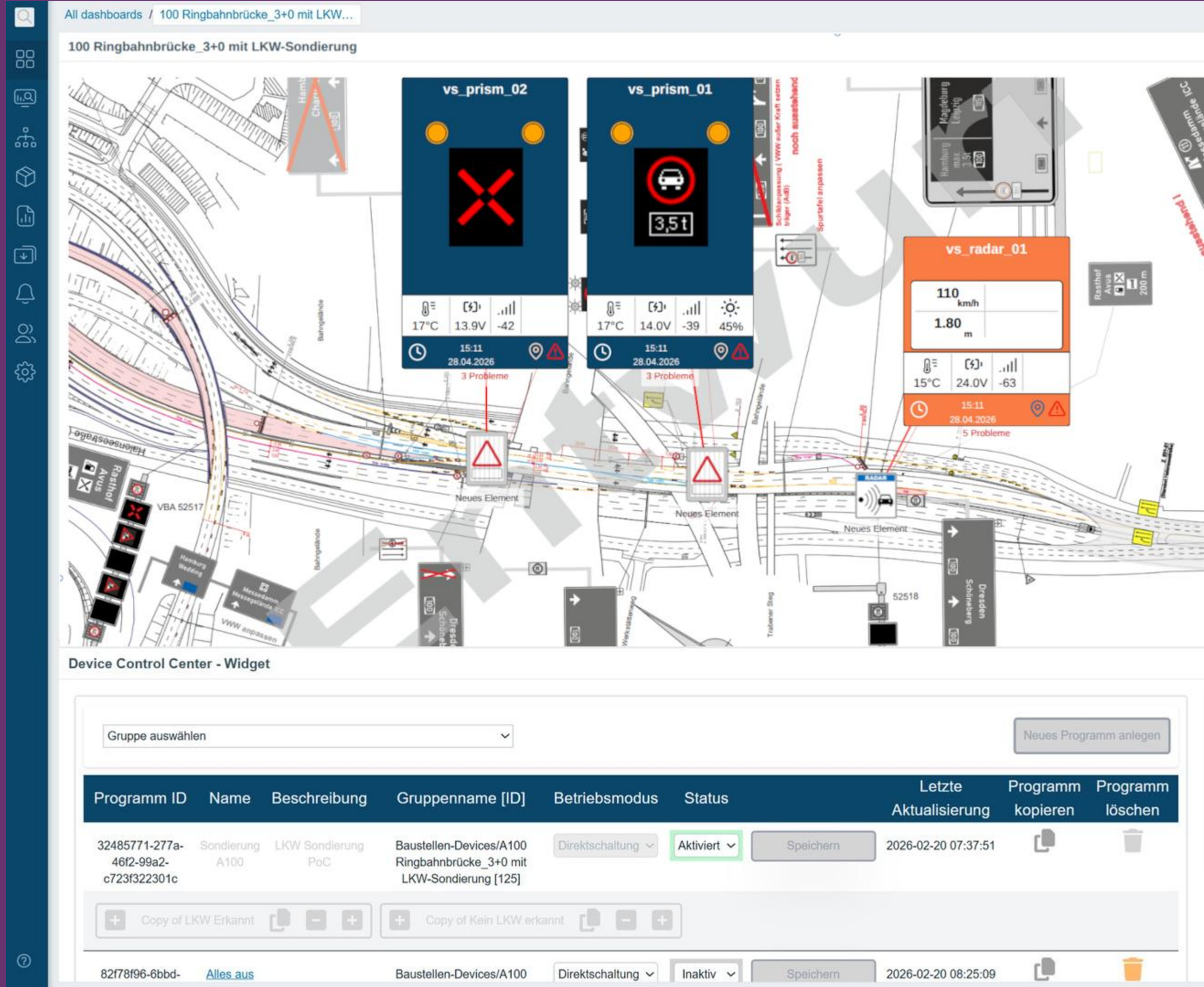
Application sample: The “Autobahn Mission”

READ RADAR VALUES AND CONTROL DISPLAY SIGNS USING MQTT

Used to dynamically control “traffic jam warning systems”

LIVE IN BERLIN (A100)

Used to dynamically re-route trucks with enhanced Radar – devices which are able to detect vehicle height next to speed.



Our vision: Extend the solution for other industries (e.g. Logistics)

1

The Invisible Pulse

- Operations in motion
- Signals remain fragmented
- No shared overview

2

The Heartbeat

- Live observability
- Trusted monitoring core
- Zabbix as heartbeat

3

The Lens

- Raw signals gain meaning
- Technical data becomes context
- Domain-specific interpretation

4

Shared Situation Picture

- One operational picture
- Shared across roles
- Context + ownership + next steps

Shared collaboration across roles, systems and operational levels

5

Software Defined Reality

- Signals become software-defined
- Derived sensors, not just raw inputs
- Across assets, systems, and people

6

Intelligent Agency

- Detect change
- Evaluate options with AI support
- Trigger the right response

7

Human in the Loop

- Governed workflows
- Visible responsibility
- Approval where needed

8

Vision Realized

- Coordinated operations
- Traceable decisions
- Continuous learning

Hilbig
IT WORKS

Thanks for attending..
and now?

The API was submitted as Zabbix Integration asset -
check it out from Github:

https://github.com/hilbigit/zabbix-graphql-api

Hilbig

IT WORKS

Mission

We are technology enthusiasts and entrepreneurs. We transform bits and bytes into solutions with real-world impact.